Raspodijeljene glavne knjige i kriptovalute

yeet

MI 2018.

1. **(8 bodova)** Razmatramo Bitcoin sustav:
   a) **(2 boda)** Koliko je dogovoreno očekivano vrijeme između blokova?
   b) **(3 boda)** Čemu služi težina rudarenja (eng. *difficulty*)?
   c) **(3 boda)** Do kojih bi problema došlo u mreži da je očekivano vrijeme rudarenja bloka 1 sekunda?

a) U prosjeku je potrebno 10 minuta
b) Težina rudarenja služi kako bi se vrijeme između blokova održavalo konstantnim. Svakih 2016 blokova se ažurira kako bi očekivano vrijeme između blokova ostalo 10 minuta. Hash bloka i nonce-a mora biti manji od praga t da bi blok bio ispravan. (veca tezina => manji prag
c) Zagušenje mreže blokovima, zatim to bi uzrokovalo problem forkanja lanca, problem je i propagacija pubkey skripta, izgubili bi optimizaciju da možemo dodati puno transakcija u blok
dodajte još problema ak netko zna -> btc je peer-to-peer mreza pa je propagacija blokova dost spora pa bi bilo vise uzaludnog rudarenja(rudaris, a ne znas da si masovno outdate-an), blockchain bi bio puno veci (i brojem blokova i podatkovno) jer bi se smanjio udio "korisnih informacija" (transakcija) u svakom bloku -> pozitivno je da bi puno prije znao dal ti je transakcija usla u blockchain (20 potvrda bi trajalo 20-tak sekundi ali trebas dosta vise potvrda jer ima vise forkova) Bilo bi problema i sa sinkronizacijom lanca, right? Jer treba vremena da čvorovi skuže da je novi blok u lancu

2. **(8 bodova)** Rudarite blokove kriptovalute koja koristi *proof-of-work* sustav. Hashrate vaše opreme je 1 Ghash/s, a čitava mreža svih rudara ima hashrate 6 Thash/s.
   a) **(1 bod)** Ukoliko je vrijeme između blokova 1 min, koliko vremena vam u očekivanju treba da izrudarite 1 blok?
   b) **(3 boda)** Što su bazeni rudara, tko njima upravlja te zašto biste se vi kao individualni rudar priključili bazenu rudara?
   c) **(2 boda)** Kako rudari u bazenu dokazuju svoj udio računalne snage uložene u rudarenje?
   d) **(2 boda)** Zašto rudar u bazenu, ukoliko pronađe *nonce* koji daje hash odgovarajuće vrijednosti, ne bi jednostavno promijenio adresu u *coinbase* transakciji i sam sebi uplatio čitavu nagradu za blok?

a) $t = \frac{1min}{\frac{1Ghash}{6Thash}} = 6000min$

b) Bazeni rudara su skupine rudara koji zajedno pokušavaju riješiti kriptografsku slagalicu i pronaći novi blok. Njima upravlja menadžer koji prima nagradu i raspodijeljuje ostalima. Priključili bismo se bazenu zbog toga što nam očekivanje nagrade ostaje isto kao i da nismo u bazenu, ali je rizik za pojedinog rudara puno manji budući da se vjerojatnost nalaženja 0 blokova drastično smanjuje. U pay-per-share modelu rudari dobivaju nagradu za svaki hash koji zadovoljava neki uvjet pa je veći rizik na pool manageru, a u proporcionalnom modelu ako bilo tko nađe blok, nagradu dijeli svaki rudar iz bazena proporcionalno radu (broju pronađenih hasheva koji zadovoljavaju neki uvjet).
c) Rudari svaki puta kada nađu hash koji zadovoljava neki uvjet (uvjet koji je puno lakše zadovoljiti nego naći hash za nagradni blok, na primjer hash s manje nula) šalju taj blok pool menadžeru. U pay-per-share modelu odmah dobivaju za takav blok fiksnu naknadu, a u proporcionalnom modelu broj blokova koji su poslali koristi se u izračunu raspodjelu nagrade kad bazen nađe ciljni hash.

d) U bloku je unaprijed kao adresa coinbase transakcije stavljena adresa pool managera koji prima nagradu i raspodjeljuje ju kasnije ostalima bez obzira tko nađe blok, budući da je blok unaprijed potpisan i transakcija postavljena na tu adresu, nakon bilo koje promjena adrese transakcija neće biti uspješno validirana
Q*: zar nije da se ionako hashira samo block header iz kojega se ni ne vide transakcije, samo njihov merkle root*? A: ~~ne, hashira se cijeli blok.~~ -- mislim da se hashira samo header, a to su merkle root transakcija, previous hash i nonce (i verzija, timestamp, i difficulty), ali merkle root nam i dalje osigurava da se ne mijenjaju transakcije C++ Znači ako bi netko htio promijenit adresu u coinbaseu promijenio bi se merkle root, i tim i dobiveni nonce koji onda više ne bi bio < t?
Tako je!

3. **(8 bodova)** Niste zadovoljni činjenicom da *proof-of-work* sustavi troše jako puno energije i razmatrate alternativne pristupe rudarenju:

a) **(3 boda)** Objasnite ukratko principe virtualnog rudarenja. Zašto bi sustavi virtualnog rudarenja bili otporni na rudarenje pomoću ASIC-a?

b) **(3 boda)** Što je *coin-age* i na koji način Peercoin implementira *proof-of-stake* principe?

c) **(2 boda)** Zašto je kod *proof-of-stake* sustava rudarima u interesu uvijek pokušavati stvoriti fork (tzv. "nothing at stake" problem)?

a) Svatko tko posjeduje valutu je rudar i plaćen je proporcionalno količini valute u sustavu koju posjeduje, odnosno može si olakšati rudarenje ulaganjem valute koju posjeduje. Potrošnja energije je puno manja i svi vlasnici valute imaju motivaciju da mreža vrijedi što više. Ovakvi sustavi su otporni na rudarenje pomoću ASIC-a jer su svi rudari jednako efikasni.

b) Coin-age je umnožak iznosa transakcije i broja blokova u kojima taj izlaz nije potrošen. Rudari si mogu prilagoditi težinu rudarenja ovisno o tome koliko žele coin-agea potrošiti, odnosno mogu uložiti puno coin-agea i malo računalne snage, ili obrnuto. Računalna snaga je ovdje primarno da se osigura slučajnost. Coin-age se nakon uspješnog rudarenja resetira. (Gdje se računalna snaga spominje u proof-of-stakeu? https://en.wikipedia.org/wiki/Proof_of_stake Jel moze neko malo pojasnit)//https://www.lopp.net/pdf/princeton_bitcoin_book.pdf // Kuzim, super hvala(233 str.) :D

c) Problem je u tome što ne postoji oportunitetni trošak. Rudar može koristiti svoj ulog za rudarenje na najduljem lancu, ali istodobno može probati kreirati fork. Zbog toga bi rudaru bilo u interesu što češće raditi forkove. Vrlo stupid pitanje ali zaš bi mi uopće pokušavali radit forkove? Jel nam to omogućuje double spending il kaj Forkovi mogu dozvoliti double spending, ali fork se može i slučajno dogoditi. Puno veći problem kod nothing at stake je što ne postoji taj oportunitetni trošak kao kod proof of work sustava kod odabira koji je fork "ispravan". Kod proof of work bi morao ulagati svoje resurse u neki od ta dva forka, a ako izabereš "krivi", onda si u gubitku. Kod proof of stake je svejedno ulažeš li u jedan ili drugi fork jer su neovisni. U biti, problem postaje kako sustav uopće može konvergirati u jedan fork na kojem svi rudari rade

Di je 4. zadatak?

4. **(8 bodova)** Baka je unuku Josipu odlučila ostaviti svoju imovinu, no htjela se pobrinuti da Josip ne dobije nasljeđe prije nego odraste. Budući da je (kao svaka moderna baka) upoznata s Bitcoinom, napisat će *locking* (*scriptPubKey*) skriptu koju će Josip moći otključati tek kada postane punoljetan, svojim potpisom i potpisom jednog od svoja dva roditelja. Josip će postati punoljetan otprilike kada će biti izrudaren blok 600.000.

   a) **(3 boda)** Napišite opisanu *locking* (*scriptPubKey*) skriptu.

   b) **(3 boda)** Napišite *unlocking* (*scriptSig*) skriput koja će otključati skriptu iz a) zadatka.

   c) **(2 boda)** Ukoliko bi baka htjela "spaliti" novce, odnosno napisati ispravnu skriptu čija sredstva nije moguće potrošiti, kako bi takva *locking* (*scriptPubKey*) skripta izgledala?

a) 600000 OP_CHECKLOCKTIMEVERIFY OP_DROP 1 pk_mama pk_tata 2 OP_CHECKMULTISIG OP_CHECKSIG
Također, ne treba li na kraju biti OP_VERIFY? Bez ovog OP_CHECKSIG
600000 OP_CHECKLOCKTIMEVERIFY OP_DROP OP_1 2 pk_josip pk_mama pk_tata 3 OP_CHECKMULTISIG

Ne može li se ovdje dogoditi da mama i tata otključaju skriptu bez josipa?DA!
link na objašnjenje
Konacno: 60000 OP_CHECKLOCsKTIMEVERIFY OP_DROP pk_josip OP_CHECKSIGVERIFY 1 pk_mama pk_tata 2 OP_CHECKMULTISIG (mislim da je ovo najbolja varijanta?)
I ovo bi vjerojatno bilo dobro rješenje, ali u prezentacijama se koristio _0 umjesto OP_1 i stavljao se u unlocking skriptu.

```
@Override
public Script createLockingScript() {
    return new ScriptBuilder()
            .number(600000)
            .op(OP_CHECKLOCKTIMEVERIFY)
            .op(OP_DROP)
            .number(2)
            .data(child.getPubKey())
            .data(dad.getPubKey())
            .data(mother.getPubKey())
            .number(3)
            .op(OP_CHECKMULTISIG)
            .build();
}
```

Prema ovome sam zakljucio da bi trebao biti OP_CHECKSIG
Je li ti onda zadnji CHECK_SIG viška pošto to već provjeri MULTISIG?
Zadnji CHECKSIG provjerava 'sig' iz unlocking skripte
Cemu OP_1 poslije OP_DROP?
Zbog toga što postoji bug u OP_CHECKMULTISIG koji uklanja dodatni element sa stacka

b) OP_0 sig_josip sig_mama(ili sig_tata)
Ne bi li ovdje trebalo **OP_0 sig_roditelj sig_josip** prema konačnom rješenju (gore)?
Da ovako treba
Zas OP_0?
Zbog toga što postoji bug u OP_CHECKMULTISIG koji uklanja dodatni element sa stacka
//Zbog čega postoji bug koji uklanja dodatni element sa stacka?
Pitaj Satoshija Nakamota
Zato sto je bug…

c) OP_RETURN

5. **(8 bodova)** Bitcoin Improvement Proposal (BIP) je dokument kojim se predlažu promjene Bitcoin standarda. BIP 34 je dodao sljedeće pravilo (malo pojednostavljeno): prvi element *scriptSig* polja svake *coinbase* transakcije mora biti visina bloka koji sadrži tu transakciju. Ova promjena je predložena kako ne bi bilo više moguće da dvije transakcije imaju isti identifikator. Recimo da se ova promjena počela primjenjivati počevši od bloka 200.000.

a) **(1 bod)** Što je indentifikator Bitcoin transakcije i kako se računa?

b) **(2 boda)** Zašto je bitno da svaka transakcija ima jedinstven identifikator? Opišite problem do kojeg dolazi ako dvije transakcije u lancu imaju isti identifikator.

c) **(2 boda)** Je li moguće da dvije *coinbase* transakcije u istom lancu u blokovima nastalima nakon bloka 200.000 imaju isti identifikator? Obrazložite odgovor.

d) **(2 boda)** Ako se BIP 34 primjenjuje od početka lanca, je li moguće da dvije obične (ne *coinbase*) transakcije u istom lancu imaju isti identifikator? Ako je moguće konstruirajte primjer, ako nije moguće detaljno obrazložite zašto.

e) **(1 bod)** Spada li BIP 34 u hard fork ili soft fork promjenu pravila? Obrazložite odgovor.

a) Identifikator Bitcoin transakcije je dvostruki SHA256 hash transakcije u poretku little endian

b) ? Ako dvije transakcije u bloku imaju isti identifikator, prilikom trošenja sredstava ulaza neće transakcija biti jednoznačno određena?
Jedino se mogu trošiti novci iz novije transakcije. Novija override-a staru. Nemoguće je doći do keša iz starije transakcije, zauvijek je izgubljen

c) Moguće je, ali je vrlo vrlo malo vjerojatno? Sto nije tu odgovor da je nemoguce jer je sha256 otporna na kolizije, a id je dvaput sha256 od transakcije, a svaka transkacija je razlicita. Da, otporna je na kolizije, ali nije nemoguće pronaći kolizije(sama definicija toga je “praktički nemoguće”) što znači da postoji neka vjerojatnost ali jako mala.

d) Rekao bih da je vrlo malo vjerojatno?
Ako je visina bloka zapravo duljina lanca, onda nije moguce jer je duljina lanca strogo rastuca funkcija
Ali uvijek postoji mogućnost da će biti isti hash, zar ne? Iznimno mala, naravno.
Kakve tu veze ima duljina lanca uopce? Duljina lanca se jedino dodaje u coinbase transakcije a ovdje je pitanje za obicne (ne coinbase) transakcije?
Mislim da je teško moguće da obične transakcije imaju isti hash. Problem s coinbase transakcijama je to što nemaju izvor, pa ako se potrefi da jedna osoba dvaput izmajna blok i treba si uplatiti točno isti iznos novca, onda defuckto postoje dvije iste transakcije, pa je to trivijalno “kolizija” (to nije def kolizije, ali to ne mijenja cinjenicu da im je isti hash). Ovo se tesko dogadja sa obicnima, one nuzno imaju nekakav izvor koji ne moze biti dvaput isti, pa je kolizija stvarno prava kolizija, a takva po definiciji sigurnosti sha256 jos nije nadjena.
(https://crypto.stackexchange.com/questions/47809/why-havent-any-sha-256-collisions-been-found-yet)

e) BIP 34 je soft fork. Čvorovi koji još nisu prihvatili novu verziju moći će nastaviti normalno raditi, no blokove koji nemaju za prvi element coinbase transakcije visinu bloka označiti će kao ispravne (ako su po svemu ostalom ispravni). Čvorovi na novoj verziji označiti će te iste blokove neispravnima.

ZI 2019 🦧🦍🍌

1. **(8 bodova)** Razmatramo Ethereum sustav:
   a) **(2 boda)** Tko sve izvršava kôd pametnog ugovora, tko plaća troškove izvođenja, a tko prima nagradu za rudarenje?
   b) **(2 boda)** Objasnite ulogu *gasa* i kako se definira nagrada za izvršavanje koda pametnog ugovora.
   c) **(2 boda)** Što je to *ommer* (uncle) blok i koja je motivacija za njegovo uvođenje u Ethereumu?
   d) **(2 boda)** Kada *ommer* blok prestaje biti ispravan i što se događa s transakcijama u njemu?

a) Zbog raspodijeljenog konsenzusa, svi čvorovi moraju izvršiti kod ugovora kako bi provjerili rezultat. Inicijator transakcije postavlja maksimalnu količinu gasa koju je spreman potrošiti i po kojoj cijeni. Nagradu za rudarenje prima rudar i računa se kao umnožak potrošenog gasa i predložene cijene.
b) Gasom se ograničava vrijeme izvođenje ugovora kao i nagrada za rudarenje. Postoji cjenik za svaku od akcija koliko troši gasa te je nagrada umnožak sume potrošenog gasa i predložene cijene.
c) Vrijeme između blokova kod Ethereuma je 12 sekundi. Zbog tako malog vremena, postoji vjerojatnost da će se pojaviti ispravni blokovi koji se neće naći konsenzusom u lancu. Njih nazivamo ommer blokovima. Svaki blok ima mogućnost referenciranja na 2 ommer bloka. U tom slučaju, rudar ommer bloka dobiva ⅞ nagrade, a rudar bloka koji ih referencira 1/32 nagrade.
d) Transakcije unutar ommer bloka se ignoriraju. Ispravni prestaju biti nakon 6 blokova.

2. **(8 bodova)** Budući da s jedne strane radite i primate plaću preko Studentskog centra, a s druge strane svakodnevno koristite usluge studentske menze, razmatrate korištenje Lightning Networka na Bitcoin blockchainu kako biste ostvarili kanal plaćanja između vas i Studentskog centra.
   a) **(3 boda)** Ukratko objasnite kako funkcionira Lightning Network i zašto bi transakcije preko Lightning Networka bile brže od onih na blockchainu.
   b) **(3 boda)** Ukoliko otvorite kanal, primite jednu plaću, 5 puta platite obrok u menzi, te nakon toga odlučite zatvoriti kanal, koliko ukupno blockchain transakcija nastane u ovom slučaju?
   c) **(2 boda)** Može li Lightning Network u potpunosti zamijeniti blockchain transakcije? Obrazložite odgovor.

a) Lightning network je protokol za plaćanje koji funkcionira na aplikacijskom sloju koji se nalazi iznad lanca blokova. Funkcionira na način da se između korisnika uspostavlja kanal i pritom korisnici daju sigurnosni depozit u zajednički novčanik. Time se stvara prva transakcija na blockchainu. Dok je kanal otvoren, korisnici mogu raditi izmjene stanja novčanika, dok god su obje stranke potpisale izmjene. U trenutku kad bilo koji od korisnika želi prekinuti kanal i dobiti svoja sredstva, to može napraviti pomoću svog privatnog ključa. U tom trenutku se stvara druga transakcija na blockchainu i dogovoreni iznosi su uplaćeni korisnicima na račun.
b) Jedine dvije transakcije na lancu blokova (i jedini troškovi) su kod uspostavljanja i

kod zatvaranja kanala plaćanja. Transakcije na LN su brže zbog toga što nije potrebno zabilježiti sve promjene na blockchainu.
c) Ne može jer se Lightning Network temelji na povjerenju između stranaka unutar kanala. Ne bi li možda čak i bilo moguće kad bi svi korisnici imali barem jedan kanal. Zbog toga što LN funkcionira na način da traži optimalan put između korisnika tijekom plaćanja, rekao bih da bi u slučaju kad su svi povezani postojao način da svatko svakome plati. Naravno, svi moraju imati dovoljno sredstva na svojim računima.To ne mjenja da se Lightning Network temelji na povjerenju dok je glavna ideja blockchain transakcija da nije potrebno povjerenje među strankama Kako se temelji na povjerenju? Nije li da ako ijedna od strana pokuša bailat da se toj strani oduzimaju sredstva koja su uložena prilikom otvaranja payment channela? Kolko ja znam glavni problem je da ova stranka koja zatvara kanal možeš koristit starije stanje kanala, i onda može npr. platit u kanalu nekome 1 ETH, onda zatvorit sa starim stanjem kanala i više-manje izvest duplo trošenje – ova druga strana koju pokušava prevarit može to riješit tak da pošalje novije up to date stanje kanala i onda dobiva sva sredstva iz kanala, ali to implicira da stalno mora gledat na blockchain da mu se ne dogodi tak nešto, to je taj problem povjerenja valjda + LN je izgrađen na blockchainu tak da nije da LN bez blockchain transakcija uopće postoji, ne može zamijenit transkacije ak ih koristi u svojoj izvedbi.

3. **(8 bodova)** Pretpostavimo da u sustavu određene kriptovalute postoje dvije nezavisne grupe rudara A i B koje implementiraju različite verzije protokola. U jednom trenutku neki napadač pronađe ranjivost u implementaciji A zbog koje će rudari prihvaćati transakcije koje dvostruko troše neki UTXO. Rudari koji koriste verziju B takve transakcije neće smatrati valjanima.

   a) **(4 boda)** Ukoliko je 80% rudarske snage na verziji A (koja sadrži ranjivost), a 20% na (ispravnoj) verziji B, što će se dogoditi s lancem blokova kad rudar s verzijom A predloži blok u kojem postoji neispravna transakcija?

   b) **(4 boda)** Što će se dogoditi u obrnutom slučaju (dakle 80% rudarske snage je na verziji B, a 20% na verziji A)?

a) većina lanca će prihvatiti blok te će s vremenom lanac koji je prihvatio blok biti duži od onog koji nije prihvatio blok zbog čega će taj blok ostati u lancu.
ovo nije istina. Nastat ce fork jer rudari koji smatraju blokove neispravnima ih nece prihvatit bez obzira sto je vecina rudara prihvatila neispravne blokove.
slažem se za ovo rješenje. ima li onda forka i u b? Realno da, ali nista dugorocno. Forkovi se mogu dogodit u B npr. ako dođe više blokova isto vremeno etc., ali kad tad će konvergirat u najduži lanac – za ovaj zadatak je ideja da će se dogodit fork na način da će se rudari A odvojit od rudara B i svatko će radit na svojem lancu.
b) većina lanca neće prihvatiti blok te će verzija koja ne prihvaća blok s vremenom postati duža zbog čega blok neće ostati u lancu.
Nastajat ce kraci forkovi (tipa onak blok ili dva ako se potrefi) od strane A rudara koji nece nikad bit prihvaceni od strane vecine. U tom slucaju se A rudarima ne isplati rudariti jer sve sta izrudare ce na kraju bit odbaceno.
zasto nece biti prihvaceno od vecina ako je 80% snage? B je obrnuti slucaj Ti neispravni blokovi nece bit prihvaceni od vecine, ispravne blokove ce svi prihvatiti.

Zato sto 80% rudara ne prihvaca te neispravne blokove, a 20% prihvaca neispravne blokove, a s druge strane svi prihvacaju ispravne blokove. I onda kad se napravi neispravni blok to ce 20% rudara prihvatit, a 80% ne i ovi od 80% ce nuzno brze rudariti iskljucivo

ispravne blokove i time uvijek imati najduzi lanac. Znaci u ovom slucaju postoji samo jedan lanac koji svi prihvacaju koji najbrze raste.
U a) zadatku najbrze raste “neispravan lanac” koji 20% rudara (B) ne prihvaca uopce i oni ce rudariti na onom lancu koji oni smatraju ispravnim, a to je lanac koji onda rudare samo i iskljucivo B rudari.

4. **(8 bodova)** Razmišljate o investiciji u kriptovalute i razmatrate na koje načine je možete ostvariti.
   a) **(4 boda)** Ako novčiće kupite na centraliziranoj burzi kriptovaluta, u kojem trenutku se transakcija između vas i prodavača zapisuje u blockchain te kriptovalute? Kad se to događa u decentraliziranim burzama?
   b) **(3 boda)** Navedite i opišite bar jedan način na koji je moguće osigurati stabilnost cijene stablecoina.
   c) **(1 bod)** Kupili ste 1 Bitcoin u 2015. godini (kad je bio znatno jeftiniji), te ste ga sad (u 2019. godini) odlučili prodati. Morate li platiti porez, i ako da, na koji iznos i po kojoj stopi?

a) Prema članku, u centraliziranoj burzi kriptovaluta transakcije se ne zapisuju na blockchain, već je burza odgovorna za održavanje sigurnosti u sustavu. Transakcije su zapisane u trenutku kada korisnik uplaćuje svoja sredstva na drugi novčanik izvan mjenjačnice. U decentraliziranim burzama se to događa kao i kod običnih transakcija.
Je li onda ovo da se kod centraliziranih burza niti ne dodaju nego se samo mijenja stanje korisnika? Dok kod decentraliziranih se provodi cijeli postupak dodavanje transakcije(zato je i sporije, ali sigurnije)?
Rekao bih da da
b) Jedan od načina za održavanje stabilnosti Stablecoina je kroz automatsko reguliranje količine stablecoina u opticaju, poznato i kao Algoritamska povezanost. Algoritamska povezanost je set pravila koja propisuje osnivač, a koja automatski u opticaj puštaju ili povlače novčiće u zavisnosti od njihove cijene. Kada je potražnja veća i cijena poraste, u optjecaj se unosi još novčića kako bi se cijena smanjila. Obrnuto vrijedi kada je potražnja manja i cijena padne.
Kako bi tu točno funkcioniralo povlačenje novaca iz optjecaja?
U nekim valutama se koristi pojam “burning” čime se novčići uništavaju. Druga mogućnost je da se ti novčići zaključaju kod njih, slično kako se zaključavaju u ETH proof-of-stake, i onda ih se otključava kako potražnja raste kako cijena ne bi bila previsoka. Ne pronalazim nigdje konkretne tehnike pa ne mogu reći sa sigurnošću.
Najjednostavniji način je da je stablecoin centraliziran, što znači da da neka institucija jamči da će im za 1 token dat definiranu protuvrijednost u fiat valuti ili čemu već. Drugi način je putem kolaterala i trezora, kao npr. Dao – da dobiješ Dao moraš založit ETH u trezor i smiješ izvadit samo određen postotak ovisno o tome koliki je kolateralizacijski omjer, oracle prati cijenu ETH na tržištu i onda ako nemaš dovoljno kolaterala netko te može likvidirat i vratit Dao natrag u trezor tako da je osigurano da sav Dao izvan trezora ima protuvrijednost unutarg tog istog i time se održava cijena.
c) Potrebno je platiti porez 10% na kapitalnu dobit. To znači da ako smo ga kupili po 5k, a prodali po 35k, moramo platiti 10% od 30k profita. Nije potrebno platiti porez ukoliko je prošlo više od 2 godine od kupnje kriptovalute. Također, moguće je sav iznos staviti u stablecoin i u njemu držati novac 2 godine pa izvući profite bez poreza
Ukratko za ovo specifično pitanje nije potrebno platiti porez
OPREZ!! Porez na kapitalnu dobit je od ove godine 10% i odnosni se na kapitalnu dobit ostvarenu u 2021, ali ako se prijavljuje porez na dobit koja je ostvarena u 2020. placa se stopa poreza od 12%!!!

(Koje je ovo predavanje?) Pravna regulativa, al nisam siguran da ima sve na slajdovima, nadam se da netko ima snimku jer je ne vidim na teamsima
Ovo je objašnjeno na jednom predavanju, no iz nekog razloga nije objavljena snimka
Možda gdpr jer je vanjski predavač?
Nije snimano to predavanje jer je predavač bio vanjski.

5. **(8 bodova)** Na drugoj stranici ispita nalazi se Solidity kôd pametnog ugovora koji omogućuje korisnicima kupnju i prodaju karata za događaje koristeći Ethereum platformu. Svi korisnici imaju pravo napraviti ponudu karata za neki događaj (pozivanjem metode `createEvent`), te bilo tko ima pravo kupiti karte iz ponude (metoda `buyNew`), dok god ima neprodanih karata. Nakon što korisnik kupi kartu on je može ponuditi po novoj cijeni (`offer`), te drugi korisnik može prihvatiti tu ponudu kako bi kupio kartu od njega (`buyOffered`).

   Nažalost ovaj ugovor je napisala osoba koja nije dobro pročitala dokumentaciju Solidity jezika te nije položila RGKK na FER-u, zbog čega ugovor sadrži nekoliko grešaka. Vaš zadatak je pronaći barem 4 greške, opisati koje su posljedice tih grešaka te predložiti kako se taj dio koda može popraviti. (Pronalazak jedne greške, opis posljedica i ispravak nosi 2 boda)

1. buyNew nece odbiti nekog tko plati premali odnos, potrebno baciti exception
2. available se izvodi nakon buyNew, trebalo bi odmah na pocetku provjeriti
3. offer potreban payable modifikator jer se gleda msg.value koji mora biti veci od transactionFee ali tu nema nikakve transakcije, zašto bi onda trebali payable? Kako sam skuzio, ako se koristi msg.value, mora ici payable kao modifikator? Nisam siguran ali ja bi rekao da msg.value mozemo pristupiti uvijek, ali kad se uplaćuju ili isplaćuju novci na/sa ugovora onda bi trebao payable. Ako sam u krivu ispravite me. Kolega je u komentarima objasnio, u dokumentaciji(https://docs.soliditylang.org/en/develop/contracts.html#function-modifiers) piše da ako se šalje ether a nema payable da se odbija transakcija. Nadalje čini se da ovaj transaction fee plaća onaj tko radi offer, iako se u trenutnom kodu taj fee ne šalje owneru nego ako samo dodamo payable bi zapeo na ugovoru. Treba onda payable ili ne? Zanemario sam činjenicu da se plaća i fee. Narančasti odgovor mi se čini točan. Mali ispravak, malo sam bolje proučio dokumentaciju, ako nije payable neće se odbiti transakcija, ali će se zanemariti ether, tako da u ovom slučaju bi jednostavno message.value uvijek bio 0 i nikad ne bi uspio offer Tl;dr: treba payable
4. offer dodati require(msg.value>transactionFee) zar ovo nije pokriveno if uvjetom već? Ovaj throw je valjda malo deprecated
5. buyOffered i buyNew ne vracaju ostatak
6. buyOffered i buyNew nece kupiti ako je cijena jednaka
7. funkcija offer - ne provjerava se nigdje jel stvarno vlasnik koji je prethodno kupio kartu želi offerat tj. ne provjerava se jel vlasnik ticketID-a za eventID i ticketID poziva funkciju offer
8. ??treba li imati konstruktor?? - da, čini se da je i to greška. trenutno postoji funkcija “ticketDepot” koja (valjda) predstavlja konstruktor, a trebala bi se zvati “constructor”, možda je i do verzije (da se prije tako pisalo.)Sumnjam da je do verzije jer bi se u tom slučaju mogao mjenjati owner ugovora aka onaj koji ubire transaction fee tako da je sigurno isto greška
   starija verzija solidity-ja je imala drugaciju sintaksu za konstruktor
9. Linija 64 i 46, send bez provjere uspješnosti. Koristiti transfer ili provjeriti da je funkcija send vratila true
10. Nedostaje funkcija kojom owner ubire svoje fees. Ili da unutar svake transakcije koja ubire fee, to direktno odmah ide owneru.

11. sha3 za offerID moze imati kolizije jer se IDjevi zbrajaju pa za razne tickete sume budu iste. Bolje bi bilo da su offerings unutar Event strukture: mapping(uint16 => Offering)
12. ticketsRemaining i numEvents su premale varijable, moze doci do overflowa.

```
pragma solidity ^0.4.0;

contract TicketDepot {

  struct Event{
    address owner;
    uint64 ticketPrice;
    uint16 ticketsRemaining;
    mapping(uint16 => address) attendees;
  }

  struct Offering{
    address buyer;
    uint64 price;
    uint256 deadline;
  }

  uint8 numEvents;
  address owner;
  uint64 transactionFee;
  mapping(uint8 => Event) events;
  mapping(bytes32 => Offering) offerings;

  function ticketDepot(uint64 _transactionFee) public {
      transactionFee = _transactionFee;
      owner = tx.origin;
    }

  function createEvent(uint64 _ticketPrice, uint16 _ticketsAvailable) returns (uint8) {
    numEvents++;
    events[numEvents].owner = tx.origin;
    events[numEvents].ticketPrice = _ticketPrice
    events[numEvents].ticketsRemaining = _ticketsAvailable;
      return numEvents; // This is eventID
  }

  modifier available(uint8 _eventID) {
    _;
    if (events[_eventID].ticketsRemaining <= 0) throw;
  }

  function buyNew(uint8 _eventID, address _attendee) available(_eventID) payable
     returns (uint16) {
    if (msg.sender == events[_eventID].owner || msg.value > events[_eventID].
       ticketPrice + transactionFee){
      ticketID = events[_eventID].ticketsRemaining--;
      events[_eventID].attendees[ticketID] = _attendee;
      events[_eventID].owner.send(msg.value - transactionFee);
      return ticketID;
    }
  }

  function offer(uint8 _eventID, uint16 _ticketID, uint64 _price, address _buyer,
     uint16 _offerWindow) {
    if (msg.value < transactionFee) throw;
    bytes32 offerID = sha3(_eventID + _ticketID);
    if (offerings[offerID] != 0) throw;
    offerings[offerID].buyer = _buyer;
    offerings[offerID].price = _price;
    offerings[offerID].deadline = block.number + _offerWindow;
  }

  function buyOffered(uint8 _eventID, uint16 _ticketID, address _newAttendee) payable {
    bytes32 offerID = sha3(_eventID + _ticketID);
    if (msg.value > offerings[offerID].price && block.number < offerings[offerID].
       deadline &&
       (msg.sender == offerings[offerID].buyer || offerings[offerID].buyer == 0) {
        events[_eventID].attendees[_ticketID].send(offerings[offerID].price);
        events[_eventID].attendees[_ticketID] = _newAttendee;
        delete offerings[offerID];
      }
  }
}
```

Sad vidim da smo možda ipak ovdje trebali pisat rješenja umjesto u komentare
Nadodao iznad slike, pogledajte ako nesto fali :)

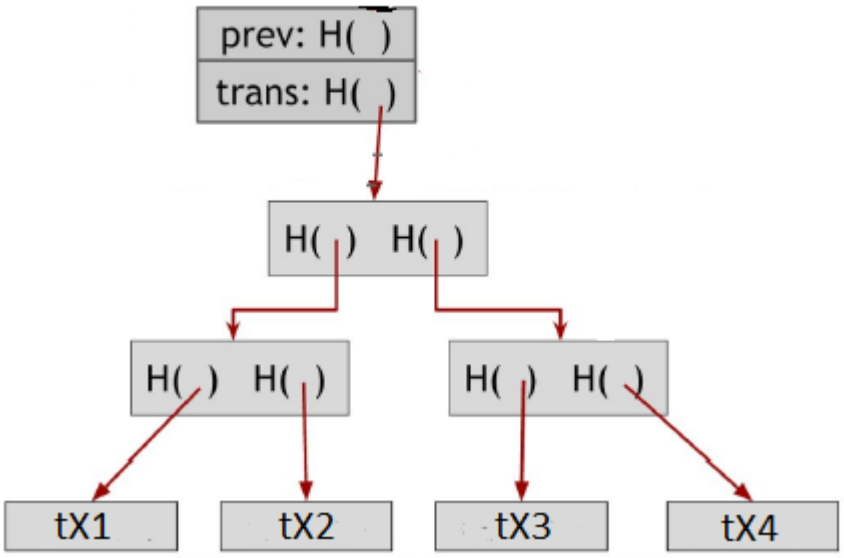
ZIR 2019.

1. **(20 bodova)** Razmatramo Bitcoin protokol konsenzusa i strukturu blokova.
   a) **(6 bodova)** Kakvu kriptografsku slagalicu rudari u Bitcoin sustavu moraju riješiti kako bi predložili novi blok?
   b) **(6 bodova)** Zašto bi rudari htjeli uključiti što više transakcija u blok koji predlažu i čime je taj broj ograničen?
   c) **(6 bodova)** Objasnite što je to Merkleovo stablo i skicirajte kako bi izgledalo Merkleovo stablo koje koristi hash funkciju $H()$ i sadrži 4 transakcije: $tx1$, $tx2$, $tx3$ i $tx4$.
   d) **(2 boda)** Zašto zaglavlje bloka sadrži korijen Merkleovog stabla transakcija?

a) Rudari moraju pronaći nonce koji zadovoljava jednadžbu:
$H(nonce|hash_{prev}|x_1|...|x_n) < t$, x_i - i-ta transakcija, t - prag

b) Rudarima je cilj ubaciti što više transakcija u novi blok jer su nagrađeni za svaku transakciju. Najveća dopuštena veličina bloka je 1MiB i time je ograničen broj transakcija.

c) Merkelovo stablo je potpuno binarno stablo u kojem svaki unutarnji čvor sadrži hash pokazivače na svoja dva djeteta.



d) Zaglavlje bloka sadrži korijen Merkleovog stabla transakcija zbog toga što će promjena ijedne transakcije u bloku uzrokovati promjenu korijena stabla, a time i promjenu hasha bloka. Čak i ako netko izmijeni sve hasheve na putu do korijena stabla od izmjenjene transakcije, ukoliko ne izmjeni vršni hash pokazivač na korijen kojem ne može pristupiti jer se nalazi u blockchainu, takav se blok ne bi mogao validirati. Dodatno, samo Merkelovo stablo se koristi da bi se slagalica koju rjesavamo zadržala na konstantnoj veličini (hasha se samo blok, koji ukljucuje hash Merkelovog stabla, ali njegova velicina ne ovisi o broju transakcija u stablu. -- malo nejasno napisano u zelenom, ja bi ovako preformulirao: hashira se samo header bloka, a header je konstante veličine, ne sadrži cijelo stablo sa svim transakcijama (kojih može biti varijabilan broj) nego samo merkle root koji je hash konstante

veličine, izmjena bilo koje transakcije automatski mijenja stablo. Blok implicira i header i merkelovo stablo s transakcijama.

2. **(20 bodova)** Želite kupiti stan i uštedili ste dovoljno Bitcoina za to, ali s druge strane imate svakodnevne životne troškove za koje su vam također potrebni Bitcoini, pa razmatrate načine na koje možete pohraniti svoje novčiće.

   a) **(6 bodova)** Što je topla pohrana (*hot storage*), a što hladna pohrana (*cold storage*)? Objasnite njihove glavne prednosti i mane.

   b) **(2 boda)** Koju od ovih metoda biste odabrali za čuvanje novaca za stan, a koju za svakodnevne troškove?

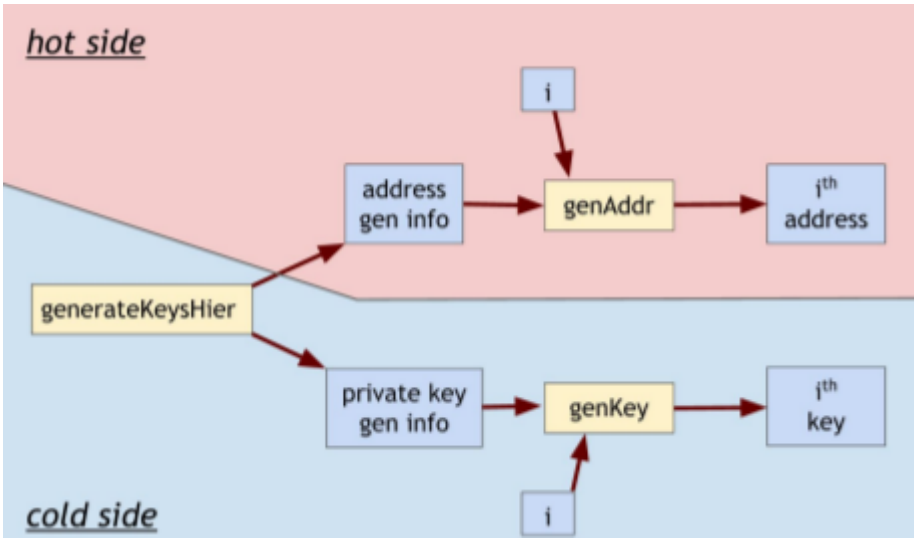
   c) **(6 bodova)** Što je to hijerarhijski novčanik? Koja je razlika u odnosu na klasično generiranje adresa (preko funkcije `generateKeys`)?

   d) **(6 bodova)** Što je to tajno dijeljenje i kako ono rješava problem *single point of failure*? Objasnite na primjeru kada je ključ podijeljen na 5 dijelova, a potrebno je znati 2 od tih 5 za rekonstrukciju ključa.

a) Topla pohrana je pohrana u kojoj su sredstva uvijek dostupna, odnosno spremljena su na nekom uređaju koji je povezan na internet. Sigurnost ovakvog spremanja je manja nego u hladnoj pohrani, no veća je praktičnost i dostupnost. Hladna pohrana je pohrana u kojoj se sredstva spremaju na fizički uređaj koji nije povezan na internet. Sigurnost ovakvog spremanja je veća, no dostupnost i praktičnost su manji.

b) Za čuvanje novaca za stan hladnu pohranu, a za svakodnevne troškove toplu pohranu.

c) Hijerarhijski novčanik je skup novčanika kod kojeg hladna strana ima proizvoljno adresa, a topla strana zna za te adrese kroz jednu kratku komunikaciju. Razlika u odnosu na klasično generiranje adresa, gdje se funkcijom generateKeys stvaraju javni i privatni ključ, je da se u ovom novčaniku generira info za generiranje javnog ključa i info za generiranje privatnog ključa. Topla strana novčanika koristi info za generiranje javnog ključa u funkciji genAddr, a hladna koristi info za generiranje privatnog ključa u funkciji genKey. Na ovaj način moguće je kontinuirano generirati parove ključeva.



d) Tajno dijeljenje je način za povećanje sigurnosti sredstava kojim se ključ dijeli na više dijelova, a samo dio tih dijelova je potreban da se inicijalni ključ rekonstruira. Ovako je povećana sigurnost jer lopov mora znati 2 od 5 ključeva da bi se domogao sredstava, a riješen je i problem single point of failure jer možemo izgubiti 3 ključa i još uvijek pristupiti svojim sredstvima.

3. **(20 bodova)** Razmatramo anonimnost i pitanje regulative kriptovaluta. Kažemo da je korisnik anoniman ako se njegove različite interakcije sa sustavom ne mogu povezati. Jedno rješenje za povećanje nepozivosti je tehnika koju zovemo miješanje (engl. *mixing*).

   a) **(8 bodova)** Zašto možemo reći da su online novčanici primjer miješanja i koji su njegovi glavni nedostaci?

   b) **(9 bodova)** Združeni novčić (engl. *coinjoin*) primjer je raspodijeljenog miješanja. Objasnite prednosti raspodijeljenog miješanja i protokol *coinjoin*-a.

   c) **(3 boda)** Koji je glavni pozitivni argument za uvođenje regulative za kriptovalute? Možete objasniti na primjeru.

a) Online novčanici nalaze se u cloudu. U cloudu se nalaz novčići i svih ostalih klijenata, tako da je moguće da u konačnici na zahtjev ne dobijemo svoje novčiće.
Glavni nedostaci su povjerenje u treću osobu, te to što nam nije osigurano miješanje. Ako se ono i događa, postoje zapisi o tome. Također često traže identitete osoba i podložni su napadima hakera. Još jedan nedostatak je da ne baratamo vlastitim ključevima, već su oni na raspolaganju stranci koja nudi online novčanik.

b) Raspodijeljeno miješanje je protokol kod kojeg više korisnika stvara jednu transakciju zajedničkim sredstvima. Svaki korisnik predaje ulazne i izlazne adresu te iznos. Transakcija miješa adrese unutar sebe te u konačnici svakom klijentu daje na uvid transakciju kako bi se potvrdila ispravnost adresa i iznosa pomoću potpisa.

c) Ponekad ishodi na tržištu nisu povoljni za većinu korisnika. Takva situacija se može riješiti regulativama. Primjer je kada se na tržištu konkurentni proizvođači dogovore oko veće cijene zajedničkog proizvoda. Takvi dogovori su nelegalni te su regulirani zakonskim okvirima. Ista mogućnost se može dogoditi i kod kriptovaluta kod prilikom manipulacijama cijenom/tečajem. Također, ukoliko se kriptovaluta regulira, cijena te kriptovalute trebala bi biti stabilnija te time privlačnija ulagačima.

4. **(20 bodova)** Pomoću Bitcoin naredbe `OP_CHECKMULTISIG` moguće je implementirati sustav prijenosa Bitcoin novčića koji osigurava obje strane u slučaju sukoba. Npr. Ante želi prodati svoj proizvod Zvonku za 1 Bitcoin. U slučaju da Zvonko nakon primanja proizvoda ne želi platiti Anti, Ante može dobiti svoje novčiće tako da neki pošteni sudac uz Antu potpiše transakciju. U slučaju da je Ante poslao proizvod koji nije ono što je Zvonko očekivao, ni Zvonko ni sudac neće htjeti potpisati transakciju pa Zvonko neće izgubiti svoje novčiće.

   a) **(8 bodova)** Napišite `pubScript` (*locking* skriptu) koja omogućava takav sustav.

   b) **(12 bodova)** Napišite `pubScript` (*locking* skriptu) za sustav u kojem postoje 3 sudca od kojih bilo tko smije potpisati transakciju u slučaju sukoba između Zvonka i Ante. Skripta mora osigurati da sudci ne mogu potpisati transakciju bez barem jednog potpisa Zvonka ili Ante.
   Napomene: Svaki sudionik u transakciji može imati više od jednog javnog i privatnog ključa. Također, u bodovanje ulazi i veličina skripte s obzirom na broj operacija i veličinu u byteovima.

a) Locking skripta: 2 pk_ante pk_zvonko pk_sudac 3 op_checkmultisig
Unlocking skripta: OP_0 sig_ante (sig_zvonko ili sig_sudac)
Nakon što potpiše Ante, treba potpisati ili Zvonko ili sudac da bi multisig prošao.

b) Locking skripta: 4 pk_ante_1 pk_ante_2 pk_ante_3 pk_zvonko pk_sudac1 pk_sudac2 pk_sudac3 7
Unlocking skripta: OP_0 sig_ante_1 sig_ante2 sig_ante3 (sig_zvonko ili sig_sudac1 ili sig_sudac2 ili sig_sudac3)
Minimalni broj potpisa svakako treba biti veći od 3 da bi osigurao da suci ne mogu sami potpisati transkaciju. Ako je minimalni broj potpisa 4, a Ante ima 3 para ključa, tako za četvrti treba potpisati ili Zvonko ili netko od 3 suca. -- ovaj odgovor vjv. nije ono što se tražilo, mislim da je ideja koristit običan op_multisig s 2 potpisa, ali ispred toga dodajemo skriptu koja provjerava da se na stogu nalazi specifično zvonko ili ante. možda najjednostavnije bi mogli npr. prvo stavit checkmultisig koja radi samo s ključem onda ante ili zvonka, a nakon toga još jednu koja provjerava da je ukupno dva potpisa, dodatno stavljamo uvjet na

unlocking skriptu da se ključevi sudaca stavljaju na kraj (skroz lijevo) ili tak nešto, eventualno dupliciramo neke ključeve ak treba i slč.
<
(sigS->sig suca, pK1 -> public key ante)
unlock: op_0 sigS sigK
lock: OP_2DUP 1 pK1 pK2 2 OP_CHECKMULTISIGVerify 2 pS1 ps2 ps3 pK1 PK2 5 OP_CHECKMULTISIG

lock: 1 <pkS1> <pkS2> <pkS3> 3 OP_CHECKMULTISIG OP_IF OP_1 OP_ELSE OP_2 OP_ENDIF <pkAnte> <pkZvonko> 2 OP_CHECKMULTISIG
unlock: OP_0 <sigAnte> <sigZvonko>
ili OP_0 <sigAnte> OP_0 <sigS(1|2|3)>
ili OP_0 <sigZvonko> OP_0 <sigS(1|2|3)>

5. **(20 bodova)** Na drugoj stranici ispita nalazi se Solidity kôd pametnog ugovora koji omogućuje korisnicima igranje igre križić-kružić. Ugovor također omogućuje korisnicima klađenje na ishod partije, tj. svaki igrač mora uplatiti neki ulog na ugovor. Svaka instanca ugovora postavljenog na Blockchain predstavlja jednu partiju između dva igrača. Za funkciju `checkGameOver()` možete pretpostaviti da je ispravno implementirana.

   Nažalost ovaj ugovor je napisala osoba koja nije dobro pročitala dokumentaciju Solidity jezika te nije položila RGKK na FER-u, zbog čega ugovor sadrži nekoliko sigurnosnih propusta. Vaš zadatak je pronaći barem 4 takva propusta koji omogućuju napadaču manipulaciju tijeka igre i/ili povlačenje novaca s ugovora bez obzira na ishod partije. Uz pronalazak propusta potrebno je opisati koje su posljedice tih propusta te predložiti kako se kôd može popraviti da više ne sadrži propust.

   Ignorirajte sintaksne greške. Pronalazak jednog propusta, opis posljedica i ispravak nosi 5 bodova. Dopušteno je/preporuča se pronalazak više od 4 propusta. Rješenja koja ne ukazuju na sigurnosne propuste i/ili još više komprimiraju sigurnost ugovora mogu biti nagrađena negativnim bodovima.

```
1  pragma solidity ^0.4.24;
2
3  contract TicTacToe {
4
5      uint32 _turnLength;
6      address[2] _players;
7
8      uint8 _p2Nonce;
9      bytes32 _p1Commitment;
10
11     uint8[9] _board;
12     uint8 _currentPlayer;
13     uint _turnDeadline;
14
15     constructor(address opponent, uint32 turnLength, bytes32 p1Commitment) public {
16         _players[0] = msg.sender;
17         _players[1] = opponent;
18         _turnLength = turnLength;
19         _p1Commitment = p1Commitment;
20     }
21
22     function joinGame(uint8 p2Nonce) public payable {
23         require(msg.sender == _players[1]);              // Check player 2.
24         require(msg.value >= address(this).balance);     // Check player 2 stake.
25         _p2Nonce = p2Nonce;                              // Register player 2 nonce.
26     }
27
28     function startGame(uint8 p1Nonce) public {
29         require(keccak256(p1Nonce) == _p1Commitment);    // Check player 1 commitment.
30         _currentPlayer = (p1Nonce ^ _p2Nonce) & 0x01;    // Select starting player.
31         _turnDeadline = block.number + _turnLength;      // Set new turn deadline.
32     }
33
34     function playMove(uint8 squareToPlay) public {
35         require(msg.sender == _players[_currentPlayer]); // Check that correct player.
36         _board[squareToPlay] = _currentPlayer;           // Make a move.
37         if (checkGameOver()) {                           // Check if game is finished.
38             selfdestruct(msg.sender);                    // Destroy contract and send
39         }                                                // founds to winner.
40         _currentPlayer ^= 0x01;                          // Set new current player.
41         _turnDeadline = block.number + _turnLength;      // Set new turn deadline.
42     }
43
44     function endGame() public {
45         if (block.number > _turnDeadline) {              // Check deadline.
46             selfdestruct(msg.sender);                    // Destroy contract and send
47         }                                                // founds to caller.
48     }
49
50     function checkGameOver() internal returns (bool gameOver) {
51         // Assume correct implementation.
52     }
53 }
```

1. 36. linija - možda nekakav require da je to polje prazno
2. 46. linija - provjeriti tko briše ugovor -> require(msg.sender==_players[0] || msg.sender==_players[1],"Only player can destruct contract") + mislim da bi trebalo isplatiti onome tko nije bio na redu jer je onom na potezu isteklo vrijeme. jel onda u liniji 46 i selfdestruct(_players[_currentPlayer xor 0x01)
3. funkcija startGame - ako se netko joina prije startGamea (dakle postavi se _p2Nonce), a nakon toga u startGameu ispadne da je _currentPlayer = 0 (odn. player1), onda on može odigrat potez pozivajući funkciju playMove (koja će promijeniti _currentPlayer u = 1 što je ispravno jer sada treba igrat drugi igrač), no on opet može pozvati startGame i tad će se opet player1 postaviti kao _currentPlayer i omogućit mu da opet igra prije drugog igrača ---- treba provjera da je igra već započela Mozemo to provjeriti da li je _turnDeadline 0? Ako je znaci da nije zapoceto - rekao bih da da. A što ako require iz 35. linije prepravimo u require(msg.sender ==

_currentPlayer) ? - _currentPlayer je int, a msg.sender je adresa. _currentPlayer predstavlja index

4. startGame se može pozvati prije nego što se drugi igrač pridružio. treba uvesti require(_p2Nonce != 0, “Player 2 didn’t join”) u startGame
5. u funkciji playMove - treba provjeriti da li je squareToPlay manji od 9? => require(squareToPlay<9) // ne, razletit ce se jer je array
6. netko joinaGame prije neko se startaoGame (dakle uplati msg.value kao i prvi korisnik), i zatim pozove endGame i pokupi ethere jer je _turnDeadline = 0 po defaulta, pa je block.number veći. => treba require(_turnDeadline != 0, "Game hasn't started yet") mozda bolje da unutar endGame bude require(checkGameOver(),”Game is not over”)? - mislim da je ispravno ovo što je pod 2., da ako je prošao deadline, onaj koji nije odigrao svoj potez je izgubio. ali ovaj require _turnDeadline zapravo brani da se pozove endGame prije početka igre.
7. board je u pocetku popunjen nulama, a igraci su 0 i 1, dakle 0 uvijek pobjeduje
8. checkGameOver ima kodomenu bool, a igra moze zavrsiti bez pobjednika -> zadnji na potezu ce pokupit novce iako nije pobjednik (komentar kaze winner takes funds)
9. konstruktor bi trebao biti payable
10. drugom igracu bi trebalo vratiti visak uloga - zasto bi ulozio vise nego prvi ako imaju jednake sanse za pobjedu

- ?gdje je ulog od prvog igraca

MI 2020.

1. Općenito
    a. Nabroji bar 3 čimbenika dobre kriptografske slagalice: ovisi o novom bloku (time i cijelom lancu), trivijalno je provjeriti njenu točnost, potrebno je mnogo računalnih resursa, podesiva težina, vjerojatnost rješavanja slagalice od nekog čvora je proporcionalna s njegovom računalnom snagom/resursima + možda bi bilo dobro spomenut ono odsustvo pamćenja?
    Mislim da je odsustvo pamćenja uključeno sa "ovisi o novom bloku" <- ne, mislim da je odsustvo pamcenja odnosi na to da ako imamo duplo vise snage imamo samo duplo vecu sansu da rijesimo. (can anyone confirm?) Pretty much. To znači da šansa da nađemo rješenje u određenom intervalu ovisi samo o količini korištenih resursa u tom intervalu
    b. Koliko traje majnanje bloka 10 min u prosjeku
    c. Zašto je rudarima u interesu stavljati transakcije u svoj blok? Čime je količina transakcija u bloku određena? jer svaka (velika većina) transakcija sadrži nagradu, količina transakcija je limitirana veličinom bloka (1MB, ja mislim)
    d. Sto je tezina slagalice i kako/zasto ju namjestamo? količina hasheva koje računalo mora generirati da riješi slagalicu, u praksi je to prag od kojeg dobiveni hash mora biti manji, namještamo ju tako da u prosjeku (očekivanju) mreža producira blok svakih 10 minuta, što je prag manji to je teže riješiti slagalicu u očekivanju Jel bi se moglo reć da je težina broj nula koji moramo dobit? Ili je to pre ne-technical? Težina ne mora biti potencija 2
2. Tri locking skripte za koje je trebalo napisati unlocking:
    a. pay to pub key hash -> <sig> <pubkey>
    b. OP_CHECKMULTISIG -> OP_0 <sig1> <sig2> ???
    c. OP_ADD OP_2 OP_EQUALVERIFY OP_SUB OP_2 OP_EQUAL -> npr 3 1 1 1
3. Rudarenje
    a. Koliko je potrebno da iskopamo blok ako imamo 0.5GHash/s a mreza ima 10THash/s t_next = 10 min / (0.5 / 10 000) = 200000 minuta
    b. Kako se mijenjaju očekivanje i varijanca ako se razmak između blokova udvostruči? Jel netko zna odgovor? Ja mislim da se očekivanje poveća 2x a varijanca smanji 2x
    u knjizi pise da se pronalazak blokova ravna prema poissonovoj distribuciji, a za nju su varijanca i ocekivanje iste vrijednosti. U nasem slucaju ta varijabla ima vrijednost broj hasheva / sekunda
    ako zelimo pronaci 1 hash u 2x vise vremena, onda one padaju 2 puta, ja sam to tako shvatio Al zaš bi očekivanje palo? Jel se to misli očekivanje t_nexta ili? Kaj ne bi to poraslo? Vrijeme između pronađenih blokova se ravna po eksponencionalnoj distribucija i varijanca je kvadrat očekivanja
    c. Koje su prednosti ulaska u bazen? Smanjuju se rizici i volatilnosti (rizik je više manje ista stvar kao volatilnost), mali rudari se mogu uključiti u posao, te je održavanje softvera lakše jer postoji nekakva centralnost.
    d. Što je sebično rudarenje? Koje sigurnosne probleme predstavlja? ~~Sebično rudarenje je kad rudar pronađe nonce, ali ne objavi blok nego nastavi dalje~~

~~rudariti na tom bloku kako bi kasnije pokupio višestruku nagradu i potratio vrijeme i resurse drugim rudarima??~~

Sigurnosni problem?? (preza 6 slajd 3 i 4 -- sebično rudarenje je privremeno zadržavanje bloka jednom kad ga pool nađe, time sebi povećava šanse da nađemo sljedeći blok prvi jer imamo prednost nad svima drugima, problem je u tome što se smanju transparentnost mreže što može dovesti do potencijalnog pada povjerenja u bitcoin, a time i vrijednosti bitcoina) Bro ja sam napisao doslovno istu stvar

4. Želimo kupiti auto za 11BTC, a imamo 4 nepotrošena izlaza transakcije (UTXO), svaki po 3 BTC.
    a. Koliko izlaznih UTXO imamo u ovoj transakciji 2 (11 btc ide prodavaču, 1 btc ide nama natrag - to su 2 izlaza; side-note: transaction fee nije dio output i on ide u iznos coinbase transakcije)
    b. Koliko locking, a koliko unlocking skripti je u ovoj transakciji? 4 unlocking, 2 locking
    c. Koliko parova locking-unlocking se mora izvršiti za validaciju transakcije? 4? yes. To je zato kaj mi moramo claimat ona 4 nepotrošena izlaza jel da? Da bismo uopće mogli uzet tih svojih 12 btc? Da, svaki input se mora otključat.
    d. Je li količina UTXO-a ograničena? Ako da, čime?
    cijeli block je max 1MB (ja mislim), a osim toga ovdje je možda poanta više da mineri ne uzimaju obzir prevelike transakcije jer su neisplative ako nemaju jako veliki fee, miner će uvijek uzeti što više transakcija jer tako maksimalno zarađuje (ako uspije riješiti slagalicu i predložiti novi blok)
5. Validacija
    a. Opiši što čvor treba napraviti kako bi validirao transakcije bloka (bar 3 uvjeta)? Postoje 4 provjere: 1) transaction validation - transaction must be valid with the current blockchain (mislim da ovo znači da za svaki input mora postojat ispravan output koji ga claima?? -- da, mislim da mora postojat svaki input koji koristimo, i transakcija mora bit sintaksno dobro složena, što se tiče outputa ne postoji neispravan output, eventualno postoji nevaljana locking skripta), 2) jesu li outputi već potrošeni ranije, 3) da transakcija već nije viđena (misli se na isti blok pretpostavljam), 4) nodes only accept and relay standard scripts based on a small whitelist of scripts. sors: knjiga
    b. Kada se blok B smatra ispravnim? kada mu je hash manji od praga t, pod uvjetom da su sve njegove transakcije validne i merkelovo stablo točno i takve stvari
    c. Hoće li čvor N zaključiti da je blok B ispravan ako sadrži hash pokazivač na blok A, a u čvoru N zadnji blok u lancu nije A? Ne kužim ovdje jel A uopće nije u lancu ili samo nije zadnji u lancu od N? Mislim da misle na čvoru N: blok B će biti ispravan samo ako mu je prethodni čvor A također ispravan (to ide rekurzivno do kraja)? Ako npr. prihvatimo blok C umjesto bloka A, koja su oba ispravna, i dolazi nam B koji je također ispravan, onda ćemo i dalje prihvatit B, jer sad on ima najduži lanac (AB: 2 > 1: C)
    d. Ako čvor odluči ne prihvatiti neki blok B, a taj blok je ispravan, hoće li ga jednom u budućnosti moći prihvatiti? Zašto? Da, ako taj blok postane dio najdužeg lanca -- npr. dođe nam blok C koji je povezan na blok B, pa tranzitivno prihvaćamo blok B? Jedini razlozi zašto ne bismo prihvatili ispravan blok su to da on nije dio najdužeg lanca ili da koristimo neku

nestandardnu strategiju rudarenja Nećemo li eventualno bit “prisiljeni” prihvatit ono što je ostatak mreže prihvatio tho? Hoćemo, da, ali to nije poanta ovog pitanja.

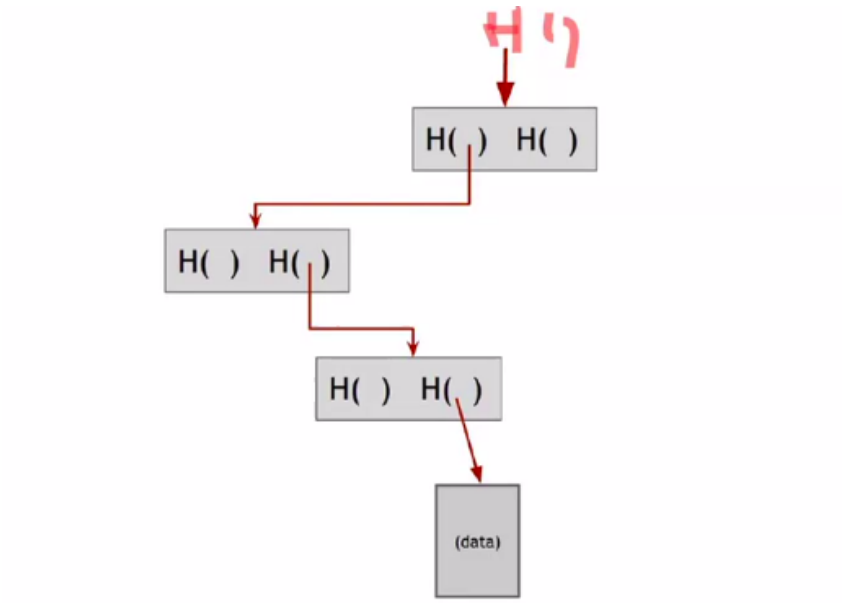
ZI 2020.

1. BTC općenito
    a. Zašto BTC transakcija nije potpuno anonimna
    b. Primjer usluge za anonimizaciju
    c. 2 razloga zašto mikrotransakcije ne funkcioniraju na BTC
    d. Objasni Lightning Network. Na koju funkcionalnost BTC se oslanja?
2. Dana tablica ETH adresa, stanja računa i nonce. Dana tablica sa transakcijama koje se izvode sekvencijalno.
    a. Koje od ovih transakcija se ne mogu izvršiti i zašto?
    b. Navedi globalno stanje za neku adresu na kraju izvođenja transakcija
    c. Ako se transakcije ne izvode sekvencijalno i istovremeno se trebaju izvesti 2 transakcije, koja će se izvesti prva i zašto?
    d. Jesu li globalna stanja na dvije adrese ista? Zašto? U čemu se razlikuju ako nisu?

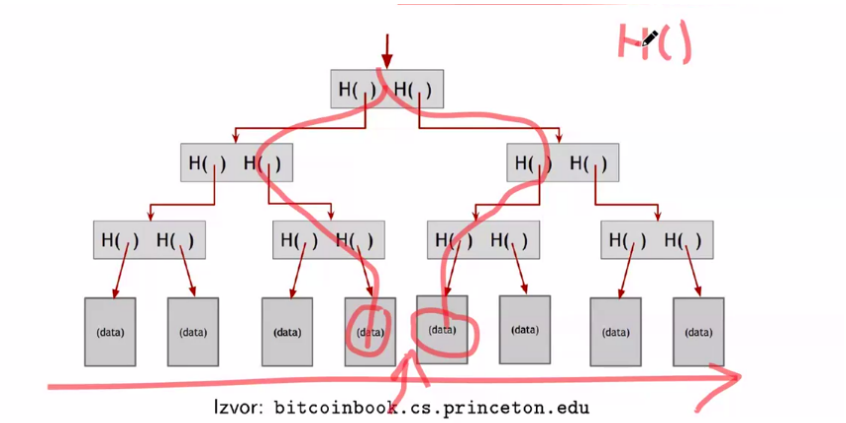
3. Random
    a. Navedi 3 kriptovalute i kako je raspodijeljena glavna knjiga ostvarena kod njih?
    b. Što je Tangle? Unutar koje kriptovalute je ostvaren? Koja je veza između izvršavanja i validacije?
    c. Što je *pump and dump*?
4. Pisac može osigurati svoje vlasništvo djela na načina da si ga pošalje na mail kad završi pisanje i ne otvori taj mail. To se zove *poor man's copyright*. nije li ovdje bilo da se šalje poštom? Kako ste odgovorili na a, b i c?
    a. Kome sud treba vjerovati ovdje da je pisac zapravo autor djela?
    b. Kako biste autorska prava ostvarili decentralizirano na BTC?
    c. Nakon nekoliko godina nestane velik dio čvorova. Je li autorsko pravo još uvijek sigurno i zašto?
    d. Navedi jednu prednost i manu ostvarenja istih svojstava na ETH vs BTC.
5. Dan je pametni ugovor King of Ether (puno jednostavnija verzija) i Logger klasa za logiranje kad netko pokuša postati kralj (uplati novce na ugovor) i kad postane kralj (kad je uplata dovoljno velika). Potrebno je pronaći i popraviti bar 4 sigurnosna propusta.

Pitanja

Kako možemo nekoga uvjeriti da se određeni list stvarno nalazi u Merkleovom stablu?
Možemo mu dati dio stabla koji hash pointerima vodi do tog određenog lista. Dakle, kažemo koje hash pointere da prati do tog lista.



Kako možemo nekoga uvjeriti da element nije dio stabla?
Sa predavanja: Prvo se mora dogovorit da ce se listovi stabla sortirati po određenom kriteriju. Zatim dati put do dva lista između kojih bi se nalazio naš list. Budući da se sortira po tom kriteriju između ta dva lista nema našeg elementa, on ne postoji u stablu.



Mislim da ovo nije baš do kraja korektno, zato što ga trebamo uvjeriti da nam je stablo stvarno sortirano. Doduše, ovo nije tako teško, možemo mu dati priliku da izbira recimo 10 indeksa elemenata, i mi mu moramo dati te puteve i oni moraju biti “sortirani”, i iznimno je nevjerojatno da njih random 10 bude sortirano, a da vlasnik stabla globalno vara. (am i missing something?)
Pa može se sam uvjerit da je stablo sortirano? Samo prolazi po svim listovima
E ali onda ima cijelo stablo. Ova sort metoda mu omogucava da ne mora dobit cijelo od vlasnika stabla.

Ima objasnjenje na teamsima predavanje RGKK:: 2. predavanje [7:34 do 10:00]
Znam da objasnjava ovaj koncept, al nije spomenuo kako dobiti povjerenje za sortiranost, a cijela stvar se temelji na dokazivanju u uvjetima nepovjerenja. Ugl mislim da nije kritično to sad za ispit, ali možda je nekom zanimljiv ovaj argument.

Što čini sustav digitalnog potpisa?

Trojka algoritama:

- $G()$ – algoritam koji generira par ključeva (sk, pk).
- $S(sk, m)$ – algoritam koji na temelju privatnog ključa sk i poruke m generira *potpis* $\sigma \leftarrow S(sk, m)$.
- $V(pk, m, \sigma)$ – algoritam koji prima javni ključ, poruku i njezin tobožnji potpis i vraća `true` ako je σ ispravan potpis poruke m odgovarajućim privatnim ključem, a `false` ako nije.

Koja su sigurnosna svojstva hash funkcije?
Otporna je na kolizije, skriva poruku, ~~korisna za slagalice~~,
//korisnost za slagalice nije sigurnosno svojstvo to nema veze sa sigurnosti to je samo poželjno svojstvo za proof of work

MI 2021.

1. Bitcoin protokol
    a. Objasniti Sybil napad, koji ga mehanizam u Nakamotovom konsenzusu sprječava
    b. Zašto je kod Pay-to-Script-Hash transakcija pošiljatelj ima manje transakcijske troškove nego za dugačke skripte?
    c. Čvor smo u Bitcoin sustavu, primili blok B koji je ispravan, hoćemo li nužno odbiti B ako nema hash pokazivač na zadnji blok u našem trenutnom lancu blokova?
    d. Što je izdvojeni svjedok (SegWit)? Dvije motivacije za uvođenje SegWita
    Izdvojeni svjedok je implementacija protokola da zaštiti protiv transakcijske malleability i poveća kapacitet bloka. ECDSA ima malleability svojstvo – moguće je promijeniti bitove potpisa, a da on još uvijek bude ispravan. Stoga, identifikator potpisane transakcije može biti različit od identifikatora transakcije koja završi u lancu. SegWit sadrži podatke koji su potrebni za provjeru validnosti transakcije ali koji nisu potrebni za određivanje učinka transakcije. Postoji dodatno Merkleovo stablo svjedoka koje odražava podatke transakcija kako bi se mogli provjeriti.
2. Ana odluči zaključati svoje novčiće lozinkom P koristeći scriptPubKey:
   `OP_SHA256 <SHA256 od P> OP_EQUAL`
    a. Napisati scriptSig koji otključava novčiće
    samo P? Da
    b. Pretpostavka: lozinka je slaba i sastoji se od 5 znakova, računanje SHA256 svih lozinki duljine 5 je u moguće u razumnom vremenu; objasniti zašto novčići mogu biti ukradeni ubrzo nakon što njena transakcija završi na lancu blokova
    c. Pretpostavka: lozinka je jaka i sastoji se od 42 znaka, je li siguran način zaštite novčića - jel netko zna dal je ovo 2^42 ? Ne nužno 2^42 nego (broj_mogucih_znakova)^42 Ovisi koliko je bitova jedan znak, onda je 2^(broj bitova jednog znaka * 42)?
3. Rudarenje blokova u Proof-of-Work sustavu. Hashrate vaše opreme 0.5 GHash/s, dok je čitava mreža 10 THash/s
    a. Prosječno vrijeme rudarenja jednog bloka je 10 min, izračunati vrijeme potrebno za pronalazak 1 bloka -> **200 000 min**
    b. Kako bi povećanje prosječnog vremena rudarenja na 20 min utjecalo na očekivanje i varijabilnost vaše zarade?
    c. Prednosti rudarenja u sklopu bazena rudara u odnosu na individualno rudarenje
    d. Što je to sebicno rudarenje? Posljedice sebičnog rudarenja na sigurnost mreže
    Sebično rudarenje jest privremeno zadržavanje bloka. Ako je rudar ispred public block chaina za 2 skrivena bloka, svo rudarenje ostatka mreže će biti bezveze potrošeno. Ostali rudari će rudariti na kraju lanca, onog koji oni misle da je najduži, ali čim netko nađe sljedeći blok, taj sebični rudar može oglasiti ta 2 bloka koja je skrivao i ovaj pronađeni blok će biti odbačen jer je ovaj drugi lanac sada najdulji.

Sebično rudarenje je uzrokovanje ostatka mreže da rasipa hash snagu pokušavajući pronaći blok koji možete odmah učiniti ustajalim, nadate se da ćete povećati svoj efektivni udio u rudarskim nagradama.

Glavni problem jest transparentnost mreže i posljedično potencijalno pad vrijednosti Bitcoina.

4. Alternativna kriptografska slagalica: treba pronaći n uzastopnih hasheva (za izračun hasheva se koristi header i n različitih nonces)
    a. Kako bi se regulirala težina rudarenja u ovakvoj slagalici? Povecanjem/smanjivanjem n
    b. Nužna svojstva kriptografske slagalice; koja svojstva nema predložena kriptografska slagalica? Slagalica nije probabilisticka (vjerojatnost rjesavanja ne ovisi o omjeru resursa u mrezi), cvor sa najvecim omjerom resursa ce uvijek prvi rjesit slagalicu
    c. Što bi se dogodilo ako se predložena slagalica ugradi u Bitcoin? Pool sa najvecim resursima ce uvijek prvi napravit blokove
    d. Glavna prednost virtualnog rudarenja u odnosu na Bitcoinov proof-of-work sustav Ekonomicnost? Okolis?
    Glavna prednost virtualnog rudarenja jest da svatko tko posjeduje valutu je rudar i plaćen je proporcionalno količini valute u sustavu koju posjeduje, odnosno, može si olakšati rudarenje ulaganjem valute koji posjeduje. Potrošnja energije je puno manja i svi vlasnici imaju motivaciju da mreža vrijedi što više. Dok kod Bitcoinovog proof-of-work sustava svaki čvor će biti dozvoljeno da se natječu jedni s drugima korištenjem svoje računalne snage. Rudari možda neće biti uloženi u dugoročno zdravlje valute. Svatko tko drži bilo koji bitcoin zapravo je dionik u valuti, a moćni virtualni rudar (kao što je onaj koji drži 51 posto ili više ukupne valute) vrlo je velik dionik. Ovaj rudar ima poticaj raditi stvari koje bi koristile sustavu u cjelini, jer takve akcije povećavaju vrijednost novčića koje drži. Ovaj argument je još jači od argumenta da se rudar koji sjedi na velikoj zalihi opreme za rudarenje čija vrijednost ovisi o budućnosti valute neće ponašati zlonamjerno.
5. Bitcoin burza ima dvije adrese, dane su statistike broja transakcija i priljev i odljev BTC-a (prva adresa 1000 transakcija, primila 20 BTC, poslala 10 BTC, druga adresa 1 transakcija, primila 10 BTC, poslala 5 BTC?)
    a. Kako se zove zapis adresa i što znači?
    b. Koja adresa pripada hladnoj, a koja toploj pohrani?
    c. Kako burza može dokazati da ukupna potraživanja klijenata ne prelazi sredstva kojima burza raspolaže na svojim adresama?